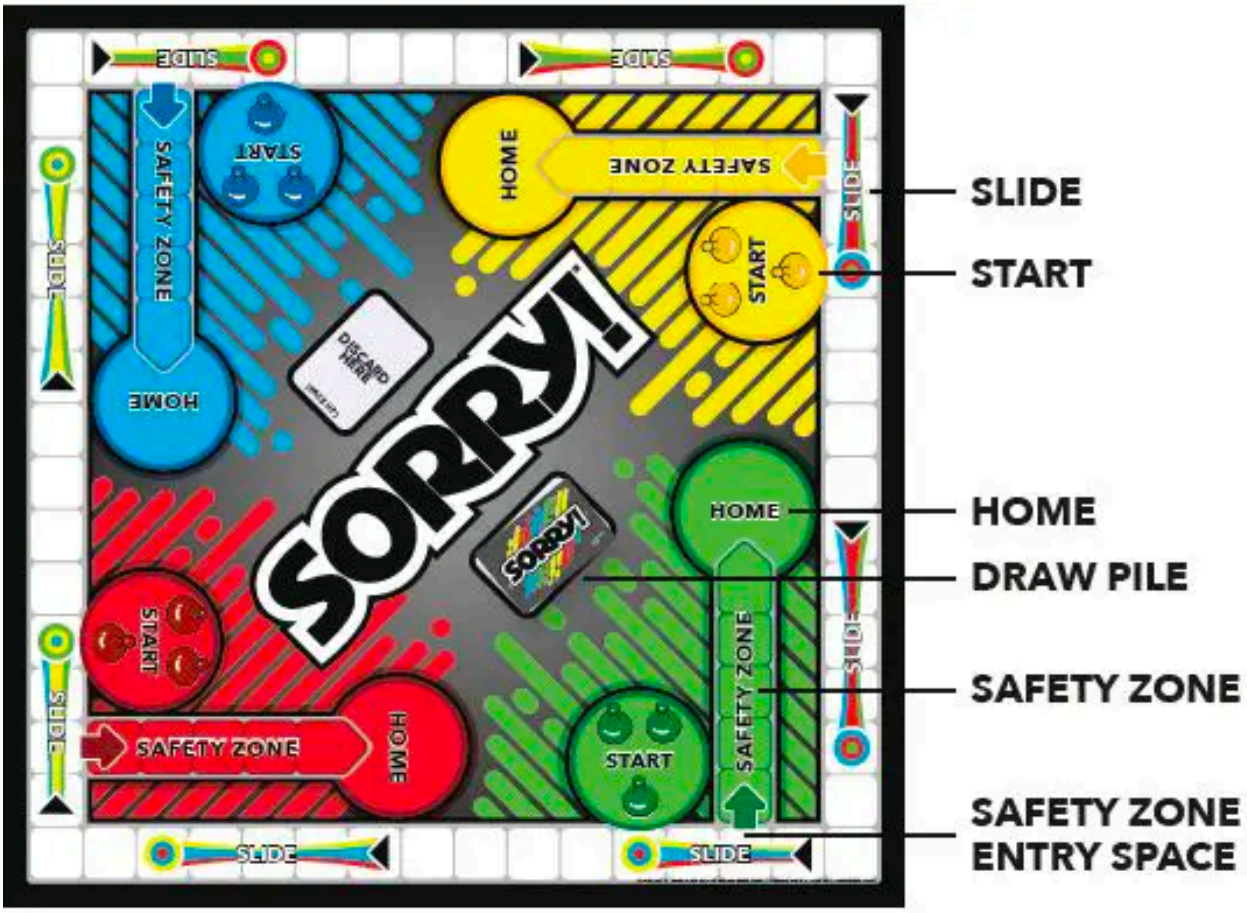
**Alma Barnhisel**
**Sorry!**




**Project 2**
**July 29, 2023**

# Introduction

## Sorry! Board Game Rules

Sorry! is a classic board game that is played with cards. The game is designed for 2 to 4 players, and the objective is to move all four of your pawns from the Start area to your Home area by following the cards' instructions and strategic movement.

Here are the basic rules of the Sorry! card game:

1. Setup:

- Each player chooses a color and places their four pawns in the Start area of their color.

- Shuffle the Sorry! cards and deal 5 cards to each player.

2. Gameplay:

- The players take turns in a clockwise direction.

- On your turn, draw a card from the deck and follow the card's instructions. The card may allow you to move a pawn out of the Start area, move a pawn forward, move a pawn backward, swap places with an opponent's pawn, or send an opponent's pawn back to their Start area.

- If you cannot make a legal move with any of your pawns, you must forfeit your turn.

3. Moving Pawns:

- Pawns can be moved clockwise around the board following the numbered spaces.

- Pawns move the number of spaces shown on the card, except for 1 or 2 spaces for a 1 or 2 card. A "Sorry!" card allows you to move a pawn from the Start area to an opponent's pawn and send the opponent's pawn back to their Start area.

- Pawns cannot pass or land on a space occupied by another pawn of the same color.

- Pawns can land on spaces occupied by pawns of other colors, which sends those pawns back to their respective Start areas.

4. Slides and Safety Zones:

- The board contains slide spaces with special colors and numbers. When you land on a slide space, you can "slide" ahead to the end of the slide, skipping the spaces in between.

- The colored Safety Zones in front of each player's Home area are safe spaces where pawns cannot be moved or sent back by opponents.

5. Home Area and Winning:

- To enter your Home area, you must draw a card that allows your pawn to move the exact number of spaces remaining to enter the Home area.

- To win, you must get all four of your pawns into your Home area.

6. Sorry! Cards:

- There are four "Sorry!" cards in the deck. These cards allow you to move one of your pawns from the Start area to an opponent's pawn and send the opponent's pawn back to their Start area. If you don't have a pawn in the Start area, you can move forward 4 spaces instead.

7. Special Moves:

- If a pawn ends its move on a space occupied by an opponent's pawn, the opponent's pawn is sent back to their Start area, and the player who landed on the space advances forward one space.

8. Winning:

- The first player to get all four of their pawns into their Home area wins the game.

The rules mentioned above provide a general overview of the Sorry! card game. The game has more detailed rules, including specific scenarios for each card type and additional variations depending on the edition or version being played. Always refer to the official rulebook or the game's instructions for complete and accurate rules.

# Summary

I was able to make a functional and basic Sorry! Game that allowed for 2-4 people to play. The game has each player take turns drawing from a deck of cards. As each player moves forward towards “Home” the program calculates which player was able to reach the final spot first. It lets the person know that they won. I utilized basic level C++ language learned in our class and will be saving the more advanced code for the second project wherein which I will include more of the Sorry! cards and Special Moves outlined above. I came to the conclusion as I was working on the second portion of this project that Sorry! Is too simplistic of a game to code and doesn’t allow for any complex code. Unlike a game that requires several different cards and can ask the user for moves, Sorry! utilizes a deck of 45 cards and has only a few opportunities for change. In order to utilize several of the necessary functions I had to use if else statements and add them all to the code and take turns using them in the game. The Sorry! Moves which would send the pawn back to the start area and the Special moves which would replace the player’s pawn and move them to the start area were able to use arrays, bubble, selection and binary sort. It was hard to implement all of the different sorting and arrays throughout the code due to the fact that the Sorry! Game did not have many player option choices.

# FlowChart Pseudo Code

```
Start
   Declare Constants
      NUM_PLAYERS = 4
      NUM_PAWNS_PER_PLAYER = 4
      NUM_CARDS = 45
      NUM_SPACES = 60
   Declare Global Variables
      cardValue = 0
   Initialize Game State
      p1Hand = Empty vector of integers
```

```
    p2Hand = Empty vector of integers
    useBubbleSort = True
    p1Pos = 0
    p2Pos = 0
Display "Welcome to the game Sorry!"
Repeat until either player's position >= 20
    Display the Game Board with player positions
    // Player 1's Turn
    Display "Player1, it's your turn. Press 'd' to draw a card: "
    Input drawIt
    If drawIt is equal to 'd' or 'D' Then:
        cardValue = drawCrd() // Draw a card with value 1 to 6
        Display "Player1 drew a card with value " + cardValue
        Add cardValue to p1Hand // Add the drawn card to Player 1's hand
        // Sort Player 1's hand using Bubble Sort or Selection Sort based on the flag
        If useBubbleSort is True Then:
            Bubble Sort p1Hand
        Else:
            Selection Sort p1Hand
        useBubbleSort = Not useBubbleSort // Toggle the flag for the next turn
        p1Pos = p1Pos + cardValue // Move Player 1 based on the drawn card
        If p1Pos > 20 Then:
            p1Pos = 20 // Cap player position to 20
        End If
    Else:
        Display "Player1, you need to draw a card. Try again."
    End If
    If p1Pos >= 20 Then:
        Display "Player1 wins! Congratulations!"
        Break from the loop
    End If
    // Player 2's Turn
    Display the Game Board with updated player positions
    Display "Player2, it's your turn. Press 'd' to draw a card: "
    Input drawIt
    If drawIt is equal to 'd' or 'D' Then:
        cardValue = drawCrd() // Draw a card with value 1 to 6
        Display "Player2 drew a card with value " + cardValue
        Add cardValue to p2Hand // Add the drawn card to Player 2's hand
        // Sort Player 2's hand using Bubble Sort or Selection Sort based on the flag
        If useBubbleSort is True Then:
            Bubble Sort p2Hand
        Else:
            Selection Sort p2Hand
        useBubbleSort = Not useBubbleSort // Toggle the flag for the next turn
        p2Pos = p2Pos + cardValue // Move Player 2 based on the drawn card
        If p2Pos > 20 Then:
            p2Pos = 20 // Cap player position to 20
        End If
    Else:
        Display "Player2, you need to draw a card. Try again."
    End If
    If p2Pos >= 20 Then:
        Display "Player2 wins! Congratulations!"
```

        Break from the loop
    End If
  End Repeat
  Display "Game Over! Thanks for playing."
End

# FlowChart

# Cross Reference for Project 1

| Chapter | Section | Topic | Where Line #'s | Pts | Notes | |
|---|---|---|---|---|---|---|
| 2 | 2 | cout | 34 | | cout << "Welcome to the game Sorry!" << endl; | |
| | 3 | libraries | 8-14 | 5 | iostream, iomanip, cmath, cstdlib, fstream, string, ctime | |
| | 4 | variables/literals | | | No variables in global area, failed project! | |
| | 5 | Identifiers | | | | |
| | 6 | Integers | 20 | 1 | int drawCrd(), plyAgn;//draw a card and play again | |
| | 7 | Characters | 44, 45 | 1 | char 'd': char 'D': | |
| | 8 | Strings | 27,28 | 1 | string p1 = "Player1"; string p2 = "Player2"; | |
| | 9 | Floats No Doubles | 20 | 1 | float dsplyBd(int p1Pos, int p2Pos);// display board, player 1 position and player 2 position | |
| | 10 | Bools | 87 | 1 | bool GameFin(int plrypos) { | |
| | 11 | Sizeof ***** | | | | |
| | 12 | Variables 7 characters or less | 19-23 | | All variables <= 7 characters | |
| | 13 | Scope ***** No Global Variables | | | | |
| | 14 | Arithmetic operators | | | | |
| | 15 | Comments 20%+ | 1-120 | 2 | Model as pseudo code | |
| | 16 | Named Constants | | | All Local, only Conversions/Physics/Math in Global area | |
| | 17 | Programming Style ***** Emulate | | | Emulate style in book/in class repositiory | |
| | | | | | | |
| 3 | 1 | cin | | | | |
| | 2 | Math Expression | | | | |
| | 3 | Mixing data types **** | | | | |
| | 4 | Overflow/Underflow **** | | | | |
| | 5 | Type Casting | 26 | 1 | srand(static_cast<unsigned int>(time(0))); | |
| | 6 | Multiple assignment ***** | | | | |
| | 7 | Formatting output | 43 | 1 | cout << p1 << ", it's your turn. Press 'd' to draw a card: "; | |
| | 8 | Strings | 28-29 | 1 | string p1 = "Player1"string p2 = "Player2" | |
| | 9 | Math Library | 97-105 | 1 | #include <cmath> | |
| | 10 | Hand tracing ****** | | | | |
| | | | | | | |
| 4 | 1 | Relational Operators | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 2 | if | 61 | 1 | if (GameFin(p2Pos))  cout << p2 << " wins! Congratulations!" << endl; | |
| | 4 | If-else | 98-100 | 1 | if (i == p1Pos && i == p2Pos) {<br>cout << "P1P2 ";<br>} else if (i == p1Pos) { | |
| | 5 | Nesting | 40-49 and 52-61 | 1 | while (!GameFin(p1Pos) && !GameFin(p2Pos)) {<br>dsplyBd(p1Pos, p2Pos);<br><br>cout << p1 << ", it's your turn. Press 'd' to draw a card: ";<br>char drawIt;//the draw input<br>cin >> drawIt;<br><br>switch (drawIt) {<br>char 'd'; | |
| | 6 | If-else-if | 38-41 | 1 | if (i == p1Pos && i == p2Pos) {<br>cout << "P1P2 ";<br>} else if (i == p1Pos) {<br>cout << " P1  ";<br>} else if (i == p2Pos) {<br>cout << " P2  "; | |
| | 7 | Flags ***** | | | | |
| | 8 | Logical operators | 24, 43, 68 | 1 | i == p1Pos and i == p2Pos | |
| | 11 | Validating user input | 47, 69, 77 | 1 | switch (drawIt) { | |
| | 13 | Conditional Operator | 91, 92 | 1 | bool GameFin(int plrypos)  return (plrypos >= 20) ? true : false; | |
| | 14 | Switch | 47, 69 | 1 | switch (drawIt) { | |
| | | | | | | |
| 5 | 1 | Increment/ Decrement | 119 | 1 | plrypos += cardVal; | |
| | 2 | While | 40 | 1 | while (!GameFin(p1Pos) && !GameFin(p2Pos)) { | |
| | 5 | Do-while | 40, 102 | 1 | do {  } while (true); // Main game loop continues indefinitely until the player chooses to quit | |
| | 6 | For loop | 115 | 1 | for (int i = 0; i <= 20; ++i) { | |
| | 11 | Files input/output both | 31,36,49, | 2 | cout << p1 << ", it's your turn. Press 'd' to draw a card: "; | |
| | 12 | No breaks in loops ****** | | | Failed Project if included | |
| | | | | | | |
| | | | | | | |
| ****** Not required to show | | | Total | 30 | Same as project 1 | |

# Cross Reference for Project 2

## You are to fill-in with where located in code

| Chapter | Section | Topic | Where Line #"s | Pts | Notes |
|---|---|---|---|---|---|
| 6 | | Functions | | | |
| | 3 | Function Prototypes | 14-25 | 4 | bool GameFin(int plrypos);<br>float dsplyBd(int p1Pos, int p2Pos);<br>int drawCrd(); |
| | 5 | Pass by Value | 14, 15, 19, 21 | 4 | • Line 14: bool GameFin(int plrypos); - int plrypos is passed by value to the GameFin function. |
| | 8 | return | 65, 88, 97, 106 | 4 | • Line 65: return (rand() % 6) + 1; - The drawCrd() function returns a random integer value between 1 and 6 (inclusive) using the modulo operator.<br>• Line 88: return plrypos >= 20; - The GameFin(int plrypos) function returns a boolean value true if plrypos is greater than or equal to 20, indicating that the game is finished. |
| | 9 | returning boolean | 78 | 4 | Line 78: bool GameFin(int plrypos) { - The function GameFin takes an integer plrypos as input and returns a boolean value. It checks if the player's position (plrypos) is greater than or equal to 20 and returns true in that case, indicating that the game is finished. |
| | 10 | Global Variables | | XXX | Do not use global variables -100 pts |
| | 11 | static variables | 176, 189, 190 | 4 | static int totalMovesP1 = 0<br>static bool seedInitialized = false; |
| | 12 | defaulted arguments | 174 | 4 | int drawCrd(int maxCard) { |
| | 13 | pass by reference | 187 | 4 | void mvPlayr(int &plrypos, int cardVal) { |
| | 14 | overloading | 73, 104, 181 | 5 | cardVal = drawCrd(1, 6); // For player 1 |
| | 15 | exit() function | 140, 141, 202 | 4 | void exitGame() { |
| 7 | | Arrays | | | |
| | 1 to 6 | Single Dimensioned Arrays | 79, 111 | 3 | Ncard[0] = p1Hand.size(); // Update the number of cards in player 1's hand |
| | 7 | Parallel Arrays | 295-306 | 2 | void mvPlayr(int playerIndex, int &plrypos, int cardVal) { |
| | 8 | Single Dimensioned as Function Arguments | N/a | 2 | Not added |
| | 9 | 2 Dimensioned Arrays | N/a | 2 | Not added |
| | 12 | STL Vectors | N/a | 2 | Not added |
| | | Passing Arrays to and from Function | 155 | 5 | void dArray(int arr[], int size) {//double array |

| | | | | | |
|---|---|---|---|---|---|
| | | Passing Vectors to and from Functions | 79, 81 | 5 | bubSort(p1Hand, p1Hand.size()); |
| | | | | | |
| 8 | | Searching and Sorting Arrays | | | |
| | 3 | Bubble Sort | 209-218 | 4 | void bubSort(vector<int> &arr, int n) { |
| | 3 | Selection Sort | 227-233 | 4 | void selSort(vector<int> &arr, int n) { |
| | 1 | Linear or Binary Search | 247-253, 261-274 | 4 | int binaryS(const vector<int> &arr, int target) { |
| | | | | | |
| | | | | | |
| ****** Not r | equired to | show | Total | 70 | Other 30 points from Proj 1 first sheet tab |

# Program Code

```
/*
 * File: main.cpp
 * Author: Alma Barnhisel
 * Created on July 29, 2023
 * Purpose:  Sorry! Board Game Version 8
 */

//System Libraries
#include <iostream>
#include <iomanip>
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <string>
#include <ctime>
#include <vector>
// Not global Constant/Sorry! Card constants :P
```

```
const int NUM_PLAYERS = 4;
const int NUM_PAWNS_PER_PLAYER = 4;
const int NUM_CARDS = 45;
const int NUM_SPACES = 60;
const int MAX_PLAYERS = 4;
int Ncard[MAX_PLAYERS] = {0}; // Number of cards in each player's hand
int ppos[NUM_PLAYERS] = {0}; // Array to store the positions of each player on the board

using namespace std;
// Function prototypes
bool GameFin(int plrypos);
float dsplyBd(int p1Pos, int p2Pos);
int drawCrd();
int drawCrd(int minVal, int maxVal); // Overloaded function void mvPlayr(int &plrypos, int cardVal);
void apgize(const string &playrN);
void bubSort(vector<int> &arr, int n);
void selSort(vector<int> &arr, int n);
int linearS(const vector<int> &arr, int target);//linear search
int binaryS(const vector<int> &arr, int target);//binary search
int cardValue;

//User Libraries
//Global Constants only
//Conversions/Math/Physics
//Execution Begins here
int main() {
    srand(static_cast<unsigned int>(time(0)));
    vector<int> p1Hand; // Player 1's hand of cards
    vector<int> p2Hand; // Player 2's hand of cards
    bool useBubSort = true; // Flag to switch between Bubble Sort and Selection Sort
```

```
string p1 = "Player1";
string p2 = "Player2";
int p1Pos = 0;
int p2Pos = 0;
int cardVal;
// Add parallel array to track player positions
int ppos[NUM_PLAYERS] = {0}; // Array to store the positions of each player on the board
//Display menu
cout << "Welcome to the game Sorry!" << endl;
do {
   // Individual game loop for each turn
   do {
      dsplyBd(p1Pos, p2Pos);
      cout << p1 << ", it's your turn. Press 'd' to draw a card: ";
      char drawIt;
      cin >> drawIt;
      switch (drawIt) {
         case 'd':
         case 'D':
            cardVal = drawCrd(1, 6); // For player 1
            cout << p1 << " drew a card with value " << cardVal << "." << endl;
            p1Hand.push_back(cardVal); // Add the drawn card to player 1's hand
            Ncard[0] = p1Hand.size(); // Update the number of cards in player 1's hand
            // Sort player 1's hand using either Bubble Sort or
            //Selection Sort based on the flag
            if (useBubSort) {
               bubSort(p1Hand, p1Hand.size());
            } else {
               selSort(p1Hand, p1Hand.size());
            }
```

```
        useBubSort = !useBubSort; // Toggle the flag for the next turn
        mvPlayr(p1Pos, cardVal);
        break;
    default:
        apgize(p1);
}

// If player1Position >= 20 Then:
if (GameFin(p1Pos)) {
    cout << (p1 + " wins! Congratulations!") << endl;
    break;
}

dsplyBd(p1Pos, p2Pos);
cout << p2 << ", it's your turn. Press 'd' to draw a card: ";
cin >> drawIt;
switch (drawIt) {
    case 'd':
    case 'D':
        cardVal = drawCrd(1, 6); // For player 2
        cout << p2 << " drew a card with value " << cardVal << "." << endl;
        p2Hand.push_back(cardVal); // Add the drawn card to player 2's hand
        Ncard[1] = p2Hand.size(); // Update the number of cards in player 2's hand
        // Sort player 2's hand using either Bubble Sort or Selection Sort based on the flag
        if (useBubSort) {
            bubSort(p2Hand, p2Hand.size());
        } else {
            selSort(p2Hand, p2Hand.size());
        }
        useBubSort = !useBubSort; // Toggle the flag for the next turn
```

```
                mvPlayr(p2Pos, cardVal);
                break;
            default:
                apgize(p2);
        }

        // If player2Position >= 20 Then:
        // Display "Player2 wins! Congratulations!"
        if (GameFin(p2Pos)) {
            cout << (p2 + " wins! Congratulations!") << endl;
            break;
        }
    } while (p1Pos < 20 && p2Pos < 20); // Individual game loop continues
    //until either player reaches position 20 or beyond
    // Reset positions and hands for a new game
    p1Pos = 0;
    p2Pos = 0;
    p1Hand.clear();
    p2Hand.clear();
    cout << "Do you want to play again? (y/n): ";
    char playAgain;
    cin >> playAgain;
    if (playAgain != 'y' && playAgain != 'Y') {
   exitGame(); // Call the exitGame function to exit the program
}
   } while (true); // Main game loop continues indefinitely until the player chooses to quit
   // Display "Game Over! Thanks for playing."
   cout << "Game Over! Thanks for playing." << endl;
   return 0;
}
```

```
bool GameFin(int plrypos) {
    return plrypos >= 20;
}
// Function to double each element in the array
void dArray(int arr[], int size) {//double array
    for (int i = 0; i < size; i++) {
        arr[i] *= 2;
    }
}

//The purpose of this function is to visually display the game board with the
//positions of both players so that the players can see their progress and
//positions relative to each other on the board.
float dsplyBd(int p1Pos, int p2Pos) {
    cout << "---------------------------------------" << endl;
    for (int i = 0; i <= 20; ++i) {
        if (i == p1Pos && i == p2Pos) {
            cout << "P1P2 ";
        } else if (i == p1Pos) {
            cout << " P1  ";
        } else if (i == p2Pos) {
            cout << " P2  ";
        } else {
            cout << "  " << i << "  ";
        }
    }
    cout << endl;
    cout << "---------------------------------------" << endl;
}
```

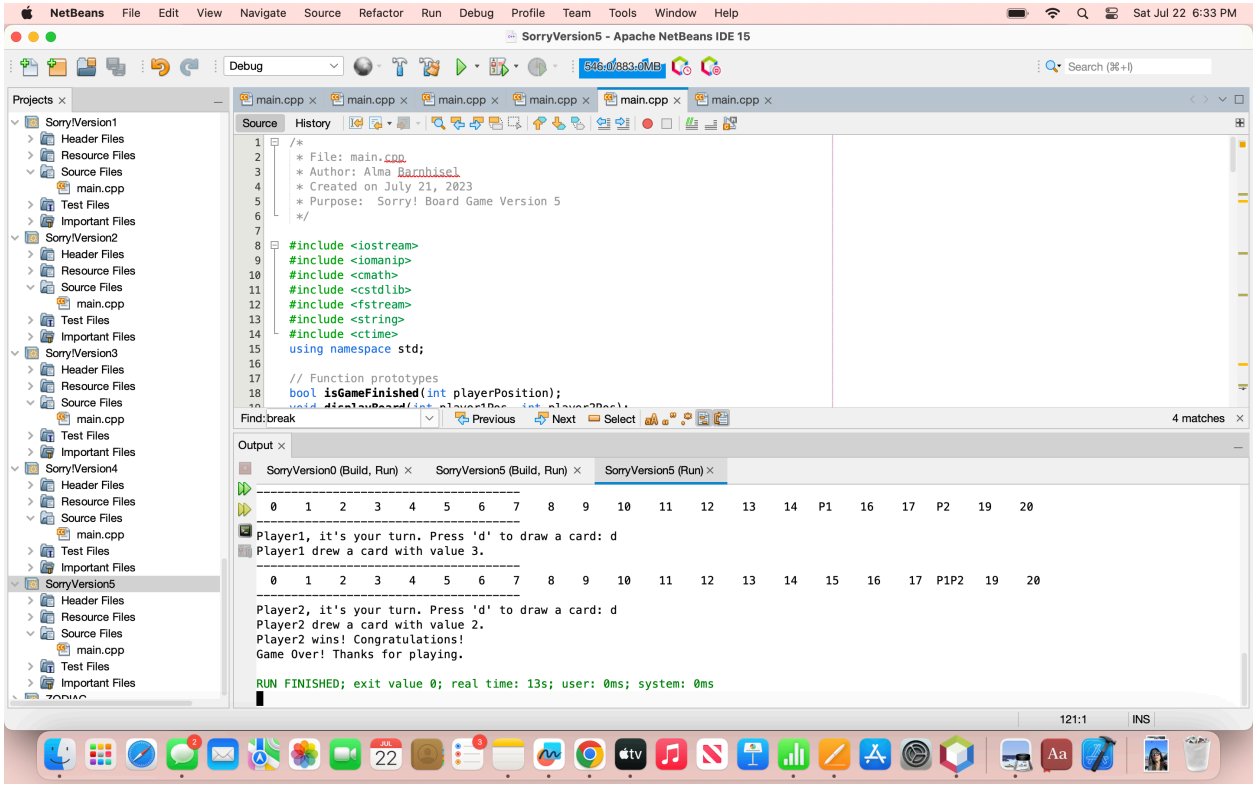
```
int drawCrd(int minVal, int maxVal) {
    static bool seedInitialized = false;
    if (!seedInitialized) {
        srand(static_cast<unsigned int>(time(0)));
        seedInitialized = true;
    }

    return (rand() % (maxVal - minVal + 1)) + minVal;
}
void mvPlayr(int &plrypos, int cardVal) {
    // Add static variables for Player 1 and Player 2 total moves
    static int totalMovesP1 = 0;
    static int totalMovesP2 = 0;

    if (plrypos > 20) {
        plrypos = 20;
    }
}

//The purpose of this function is to display a message to the player, prompting
//them to draw a card and try again when they provide an invalid input or fail to
//draw a card during their turn in the Sorry! game.

void apgize(const string &playrN) {
    cout << playrN << ", you need to draw a card. Try again." << endl;
}
void exitGame() {
    cout << "Exiting the game. Goodbye!" << endl;
    exit(0); // Terminate the program with exit code 0 (success)
}
```

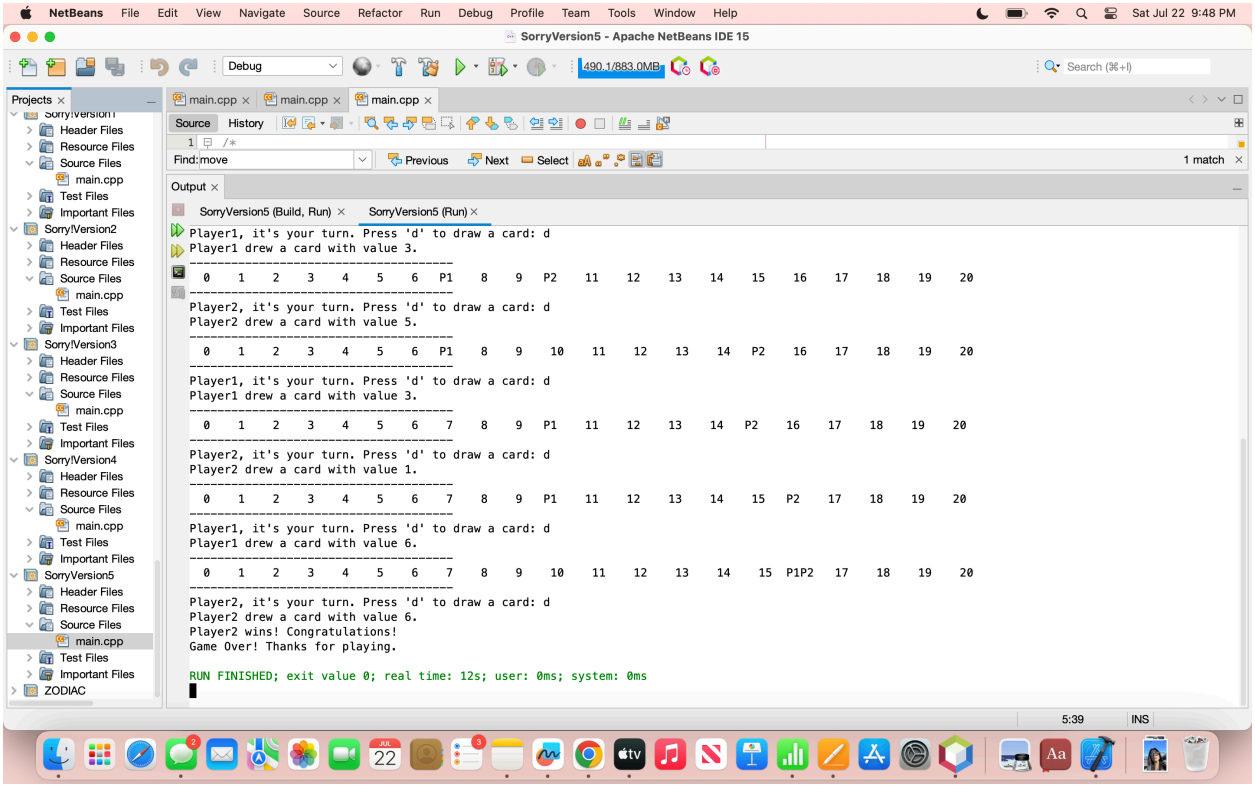
```
//In the context of the Sorry! game, we are using Bubble Sort to sort the
//player's hand of cards in ascending order, so they can easily see and manage
//their cards during the game.
void bubSort(vector<int> &arr, int n) {
    // Implement bubble sort algorithm
    for (int i = 0; i < n - 1; ++i) {
        bool swapped = false;
        for (int j = 0; j < n - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                swap(arr[j], arr[j + 1]);
                swapped = true;
            }
        }
        if (!swapped) {
            break;
        }
    }
}
//To use both Bubble Sort and Selection Sort at the same time, we can alternate
//between the two sorting algorithms for sorting the player's hand of cards
//in the Sorry! game.
void selSort(vector<int> &arr, int n) {
    // Implement selection sort algorithm
    for (int i = 0; i < n - 1; ++i) {
        int minIndex = i;
        for (int j = i + 1; j < n; ++j) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
```

```
        if (minIndex != i) {
            swap(arr[i], arr[minIndex]);
        }
    }
}
// Linear Search function implementation
//In this code, I've added the linearSearch function, which performs a linear
//search to find the target value in the array. If the target value is
//found, it returns the index where it is found; otherwise, it returns -1
//to indicate that the target value is not present in the array.
int linearS(const vector<int> &arr, int target) {
    for (int i = 0; i < arr.size(); ++i) {
        if (arr[i] == target) {
            return i; // Found the target value at index i
        }
    }
    return -1; // Target value not found in the array
}
// Binary Search function implementation (Assumes the array is sorted in ascending order)
//the binarySearch function, which performs a binary search on a sorted array.
//This function assumes that the input array is sorted in ascending order.
//If the target value is found, it returns the index where it is found;
//otherwise, it returns -1 to indicate that the target value is not present in the array.
int binaryS(const vector<int> &arr, int target) {
    int left = 0;
    int right = arr.size() - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target) {
            return mid; // Found the target value at index mid
```

```
        } else if (arr[mid] < target) {
            left = mid + 1; // Discard the left half of the array
        } else {
            right = mid - 1; // Discard the right half of the array
        }
    }
    return -1; // Target value not found in the array
}
void mvPlayr(int playerIndex, int &plrypos, int cardVal) {
    // Add static variables for Player 1 and Player 2 total moves
    static int totalMovesP1 = 0;
    static int totalMovesP2 = 0;
    if (plrypos > 20) {
        plrypos = 20;
    }
    // Update player's position based on the card value
    ppos[playerIndex] += cardVal;
}
```

# Output


NetBeans File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Sat Jul 22 6:33 PM
SorryVersion5 - Apache NetBeans IDE 15
Debug
Projects
Sorry!Version1
Header Files
Resource Files
Source Files
main.cpp
Test Files
Important Files
Sorry!Version2
Sorry!Version3
Sorry!Version4
SorryVersion5
main.cpp main.cpp main.cpp main.cpp main.cpp main.cpp
Source History
/*
* File: main.cpp
* Author: Alma Barnhisel
* Created on July 21, 2023
* Purpose: Sorry! Board Game Version 5
*/
#include <iostream>
#include <iomanip>
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <string>
#include <ctime>
using namespace std;
// Function prototypes
bool isGameFinished(int playerPosition);
Find: break
Previous Next Select
4 matches
Output
SorryVersion0 (Build, Run)
SorryVersion5 (Build, Run)
SorryVersion5 (Run)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 P1 16 17 P2 19 20
Player1, it's your turn. Press 'd' to draw a card: d
Player1 drew a card with value 3.
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 P1P2 19 20
Player2, it's your turn. Press 'd' to draw a card: d
Player2 drew a card with value 2.
Player2 wins! Congratulations!
Game Over! Thanks for playing.
RUN FINISHED; exit value 0; real time: 13s; user: 0ms; system: 0ms
121:1
INS


NetBeans File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Sat Jul 22 9:48 PM
SorryVersion5 - Apache NetBeans IDE 15
Debug
490.1/883.0MB
Search (⌘+I)
Projects
main.cpp
main.cpp
main.cpp
Source
History
Find:move
Previous
Next
Select
1 match
Output
SorryVersion5 (Build, Run)
SorryVersion5 (Run)
Player1, it's your turn. Press 'd' to draw a card: d
Player1 drew a card with value 3.
0 1 2 3 4 5 6 P1 8 9 P2 11 12 13 14 15 16 17 18 19 20
Player2, it's your turn. Press 'd' to draw a card: d
Player2 drew a card with value 5.
0 1 2 3 4 5 6 P1 8 9 10 11 12 13 14 P2 16 17 18 19 20
Player1, it's your turn. Press 'd' to draw a card: d
Player1 drew a card with value 3.
0 1 2 3 4 5 6 7 8 9 P1 11 12 13 14 P2 16 17 18 19 20
Player2, it's your turn. Press 'd' to draw a card: d
Player2 drew a card with value 1.
0 1 2 3 4 5 6 7 8 9 P1 11 12 13 14 15 P2 17 18 19 20
Player1, it's your turn. Press 'd' to draw a card: d
Player1 drew a card with value 6.
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 P1P2 17 18 19 20
Player2, it's your turn. Press 'd' to draw a card: d
Player2 drew a card with value 6.
Player2 wins! Congratulations!
Game Over! Thanks for playing.
RUN FINISHED; exit value 0; real time: 12s; user: 0ms; system: 0ms
5:39
INS